软件设计文档

目　　录

1. 开发规划

1.1 开发人员

角　色	主要职责	负责模块	人员	贡献度
项目经理 PM	■ 项目全面负责 ■ 项目设计 ■ 主要框架/模块编写 ■ 项目进度控制 ■ 项目架构设计	■ App 应用端“我的”模块	林子博	30%
产品经理 PT	■ 定义需求 ■ 产品监督 ■ 结果验证（测试） ■ 用户文档 ■ 项目架构设计	■ 服务器后台模块开发	庄清惠	30%
程序员 DEV	■ 代码编写 ■ 软件需求规格说明书	■ App 应用端“推荐”模块	张双涛	20%
程序员 DEV	■ 代码编写 ■ 项目开发计划文档 ■ 软件设计文档	■ App 应用端“资讯”模块和“更多”模块	林得堡	20%

1.2 开发计划

<附项目开发计划>

1.3 开发环境和工具

开发工具：Webstrom, Android Studio, Apache, Mongodb

1.4 开发规范

每个人的编码习惯和风格都不同。定义好规范，才能统一风格，才可提高代码可读性，同时也提高了维护性，还减低了引入 bug 的机会。开发规范并没有统一的标准，在项目中，我们使用下面的几点：

缩进

很多人都习惯用 Tab 缩进，统一设置好 Tab 缩进的 size=4 就好了。

命名

一个好的命名，一眼就可以从名字中看到它是干嘛的，做什么用，什么类型等等。举个 id 命名的例子：功能控件（驼峰式），例如：collectBtn，这是一个关于收藏的按钮。

单位

文字大小的单位应该统一用 sp，其他元素用 dp。因为这两个单位是与设备分辨率无关的，能够解决在不同分辨率设备上显示效果不同的问题。

2. 总体设计

2.1 运行环境

服务器端：Apache Tomcat

2.2 开发环境支持

2.2.1 硬件配置要求

- 网络：2G、3G、4G 网络
- 机器配置：CPU 需 1.5GHz 或以上，内存 512M 以上
- 内存：2G 或以上内存或 SD 卡

2.2.2 软件技术支持

- 实现语言：Java, JavaScript；
- 系统支持：Android 4.2 及以上、Android Studio、 Apache Tmocat 6.0、WebStrom
- 文档管理和版本控制：GIthub
- 建模工具：Enterprise Architect 8、 Matlab
- 界面设计工具：Android Studio
- 其他软件支持：Adobe Photoshop 6

2.3 系统结构

2.3.1 用户表示层（UI）

表示层主要包含 Windows 窗体、用户界面等元素。该层主要完成两个任务：一是从业务逻辑层获取数据并显示给用户；二是实现与用户的交互，将有关数据回送给业务逻辑层进行处理，其中可能包括数据验证、处理用户界面事件等。

表示层的价值在于，它把业务逻辑层和外部刺激（用户输入、激发事件等）隔离开来。这样到达业务逻辑层的请求看起来是一样的，无论请求是来自用户输入，还是接受一个文件、时间或者业务事件等所触发。另外，表示层重点表达的是用户的兴趣和利益，为应用程序交互提供任何形式的帮助，包括有益的信息提示、用户偏好设置等。

2.3.2 业务逻辑层（BLL）

业务逻辑层（Business Logic Layer）无疑是系统架构中体现核心价值的部分。它的关注点主要集中在业务规则的制定、业务流程的实现等与业务需求有关的系统设计，也即是说它是与系统所应对的领域（Domain）逻辑有关，很多时候，也将业务逻辑层称为领域层。业务逻辑层在体系架构中的位置很关键，它处于数据访问层与表示层中间，起到了数据交换中承上启下的作用。

2.3.3 数据访问层(DAL)

数据访问层包含数据存储和与它交互的组件或服务。这些组件和服务在功能上和业务逻辑层相互独立。其功能主要是负责数据库的访问，可以访问数据库系统、二进制文件、文本文档或是 XML 文档。

2.4 限制和约束

1) 性能（Performance）

能够完成人们所期望的工作。

2) 可靠性（Reliability）

系统在应用或错误面前，在意外或错误面前使用的情况下维持软件系统功能特性的基本能力。

3) 可用性（Availability）

指系统能够正常运行的时间比例，可用两次故障之间时间的长度或者出现故障时系统能够恢复正常的速度来表示。

4) 健壮性（Robustness）

模块应能够承受一定的压力，保证 24 小时无故障运行。

5) 安全性（Security）

报文中的关键数据域以密文的方式传输，可分为机密性、完整性、不可否认性和可控性等特性。

6) 可修改性（Modifiability）

能够快速地以较高的性能价格比对系统进行变更，通常也叫可维护性。

7) 可变性（Variability）

该系统结构能够经扩充或变更为新体系结构。

8) 易用性（Usability）

操作界面通俗易操作。

9) 可扩展性（Expansibility）

应该充分考虑到将来交易的修改或增加，避免需求变更时大规模修改程序；

10) 互操作性（Interoperability）

系统与外界或系统与系统之间能够进行交互操作。

3. 数据库设计

<附数据库设计文档>

4. 接口规范

4.1 用户接口

A．互联网

通过互联网，与服务器进行交互。实现查询和记录功能。

B．手机 GPRS

通过手机 GPRS 连接到互联网或在本地进行查询与记录。

4.2 外部接口

电信、移动、联通运营商，互联网

4.3 内部接口

在不同层与不同模块之间主要通过实体对象来实现通信。将每个实体对象的所有属性封装到不同实体类中，通过实体类在业务层与表现层之间通信。业务层与数据层之间通过调用存储过程来实现。

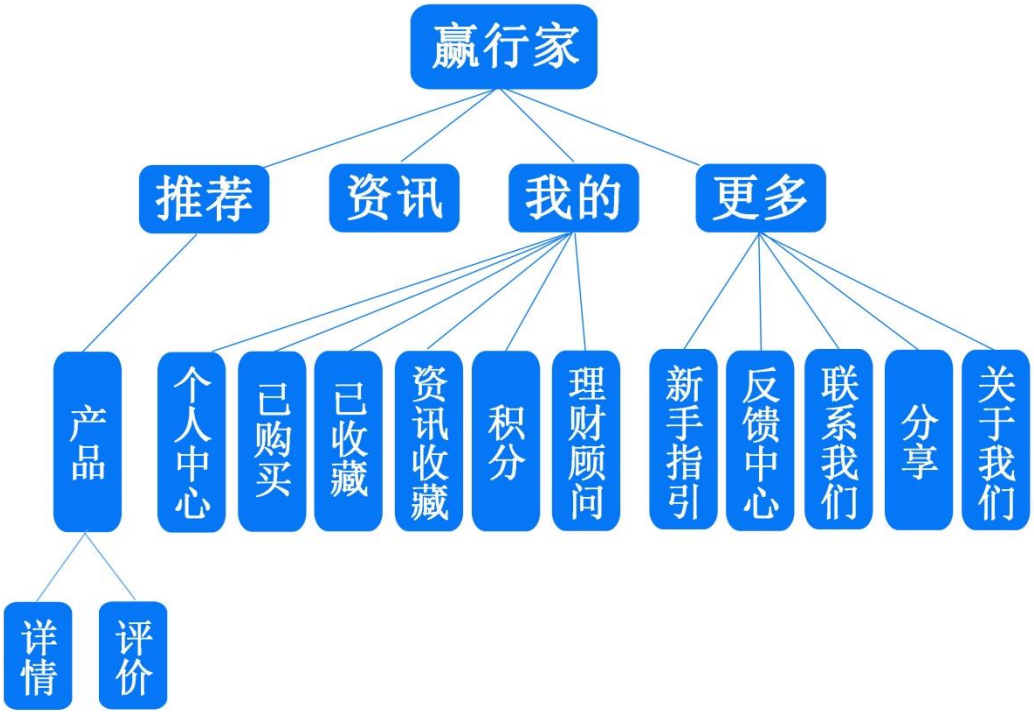
在每个模块的每个实体内部则通过类属性来实现。通过减少使用全局变量来提高程序的复用性，降低程序的耦合度。

<附接口设计文档>

5. 系统结构设计

5.1 系统总体架构

系统功能结构图：



5.2 逻辑结构设计要点

5.1 业务实体

业务实体是代表业务数据的对象，是业务逻辑层的基础。在应用程序代码中，业务实体表现为一些类。这些类在实现过程中，定义了私有字段、公共属性、构造函数等。这些结构正是创建典型业务实体类的主要成员。

- 私有字段

用于高速缓存本地业务实体的数据。当通过数据访问组件从数据库中检索数据时，这些字段保留数据库中数据的一个快照。

- 公共属性

用于访问实体状态、访问实体内部数据的子集和层次（子集和层次可使用类属数组来实现）。属性可以和数据库中的列同名，但不是必需的。开发人员可根据需要选择属性名称，而不是单纯以数据列的名称为依据。

- 构造函数

为业务实体类的初始化奠定基础。对于业务实体组件而言，通常都存在多种构造函数。另外，在业务实体类中还可能包括方法、事件和属性等。

- 方法

通过使用业务实体组件中的数据，执行本地化处理。

- 事件

实现业务实体组件内部状态的变化。

业务实体具有以下特征：

- 业务实体提供了对业务数据和相关功能的状态化编程访问。
- 业务实体可以使用具有复杂架构的数据来构建。这种数据通常来自数据库中的多个相关表。
- 业务实体数据可以作为业务逻辑组件方法中的输入/输出参数进行传递。
- 业务实体通过序列化，保持它们的当前状态。如：应用程序可能需要在本地磁盘、桌面数据库（如果应用程序脱机工作）或消息队列中存储实体数据。
- 业务实体不启动任何类型的事物处理。事物处理由使用业务实体的应用程序或业务过程来启动。

5.2.2 实现业务实体

在实现业务实体组件之前，尤其是实现复杂业务实体之前，必须经过两个步骤：一是实现逻辑映射，即在逻辑上，实现将关系数据映射到业务实体。二是对业务实体进行编码。

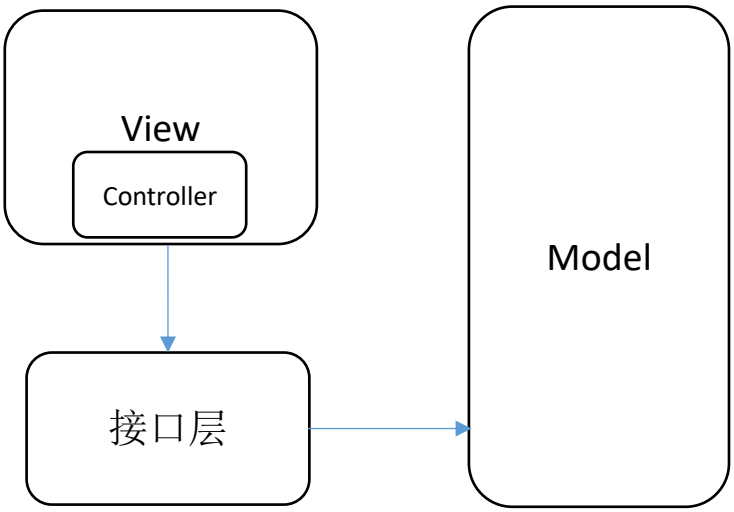
- **实现逻辑映射的建议**
  - 多花时间分析应用程序的应用需求和逻辑业务实体，然后为之建立模型，接着考虑业务实体的创建问题，而不要为每个数据表都定义单独的业务实体。建立应用程序的工作方式模型的方法之一是使用统一建模语言 UML。
  - 不要定义单独的业务实体来表示数据库中的多对多表，可以通过在数据访问组件中实现的方法来公开这些关系。
  - 如果需要实现返回特定业务实体类型的方法，则建议把这些方法放在该类型对应的数据访问组件中。
  - 数据访问组件通常访问来自单一数据源的数据。当需要聚合多个数据源的数据时，建议分别为访问每个数据源定义一个数据访问逻辑组件，这些组件可以由一个能够执行聚合任务的更高级组件来调用。

6. 软件技术选型

6.1 APP 应用端采用 MVC 模型

选型原因：

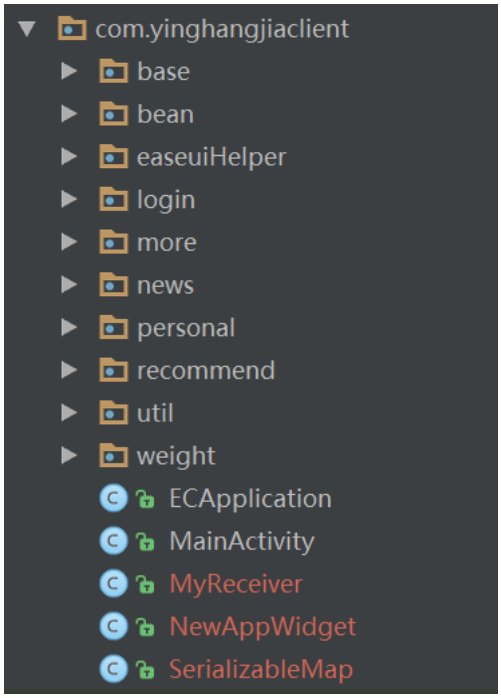
由于我们的应用主要的功能在于信息展示（包括理财产品信息、新闻信息），功能比较简单，所以在考虑架构时，就使用比较熟悉的 MVC 模型。而使用的 MVC，与一般意义的 MVC 还有所不同，鉴于业务简单，没有复杂的逻辑操作,但是开发周期较短,所以我们将 Controller 层与 View 层合并在一起，牺牲了一定的扩展性用于加速开发，但是在进行 View 和 Controller 层融合过程，也注意到类内的模块化，所以然后之后有项目重构的需要，抽取出 Controller 也是比较容易的。除了 MVC 之外，我们引入一个接口层，接口层封装了服务器提供的 API，以及一些通用的操作工具。



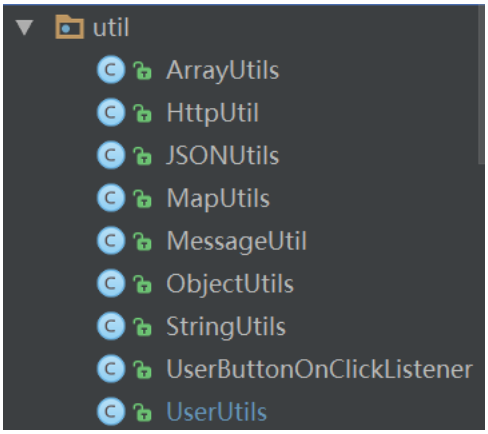
总体架构：

使用了融合 Controller 和 View 层的快速开发 MVC 架构，如上图

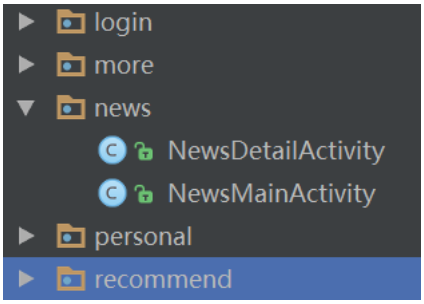
包结构：



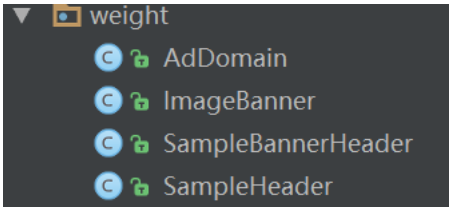
1）其中的 utli 属于接口层，所有的接口命名为 xxUtils



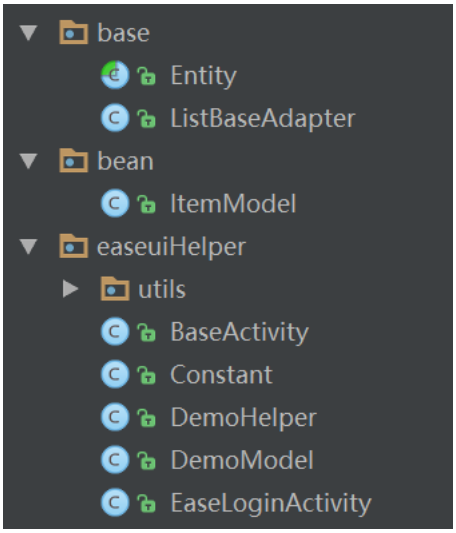
2）其中的 Login、More、News、Personal、Recommend 为 View+Controller 层，细分下去的话，就是每一个功能模块下面都是一系列子界面，每个子界面就是一个 xxActivity。



3）其中 Weight 为界面元素的 model 层



4）Base、Bean、EasuiHelper 为例外，这三个包是引入环信移动客服相关功能而使用的模块。

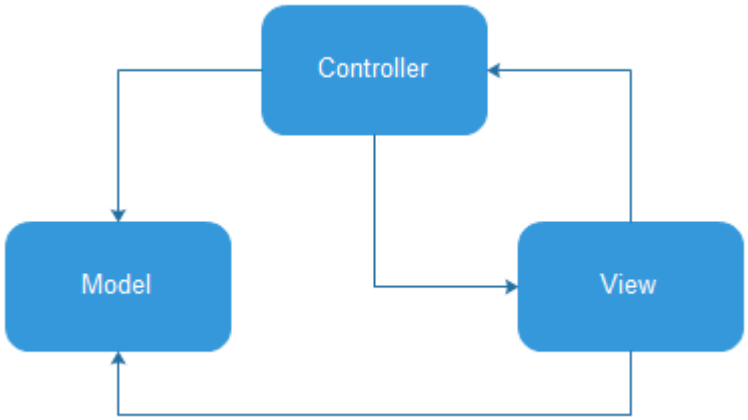


模块划分：

我们根据按功能进行划分，这样的划分有一个好处，由于在项目的设计过程中，有着诸多的需求。而很多需求都可以进行归类，根据功能需求分类的方法进行模块的划分。可以让需求在归类上得到明确的划分，而且通过功能需求进行软件的模块划分使得功能分解，任务分配等方面都有较好的分解。具体的模块划分依据：在 App 中导航栏四个功能块，都划分为一个模块包，再加上一个登陆注册模块，一共就是五个模块，每个模块下面包含了功能块所属的多个子界面。因为使用了环信移动客服 SDK，所以多出了三个看起来有些另类的模块 Base、Bean、EasuiHelper

6.2 服务端采用 MVC 模型

MVC 全名是 Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC 被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。



具体的层次结构如下：

Model（模型）	应用程序中用于处理应用程序数据逻辑的部分 通常模型对象负责在数据库中存取数据
View（视图）	应用程序中处理数据显示的部分 通常视图是依据模型数据创建的
Controller（控制器）	是应用程序中处理用户交互的部分 通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

后端主要分为 service，controller，model 三块。Model 负责定义数据库模型。Service 提供 model 之上的操作。Controller 负责路由和请求的处理。

利用 MVC 模式，在分层的同时也简化了分组开发。不同的开发人员可同时开发视图、控制器逻辑和业务逻辑，加快了开发的速度。

在服务端上，我们使用的是近年来兴起的 node.js。node.js 的诞生就是为了编写高性能的服务器，同时接入繁荣的 JavaScript 生态圈。node.js 的开发

有极其丰富的资源。而极具特色和灵活性的 JavaScript 语言也给这门技术增添了很多魅力。node.js 本身异步非阻塞的特点，使得它非常适合做服务端的开发。能够轻松做到非常大的 IO 吞吐量。这就是我们选择 node.js 的原因。

服务端选用的开发框架是 Koa。Koa 是著名 web 框架 express 原班人马打造的下一代 web 框架。更加小巧，灵活和健壮。它的设计理念是嵌套式的中间件。同时推荐使用 async/await 这个新的语法特性，使得 node.js 异步编程更加简单。

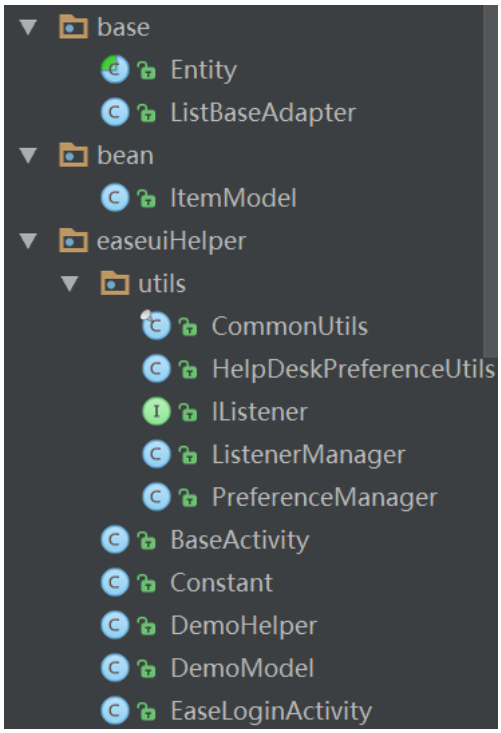
6.3 数据库

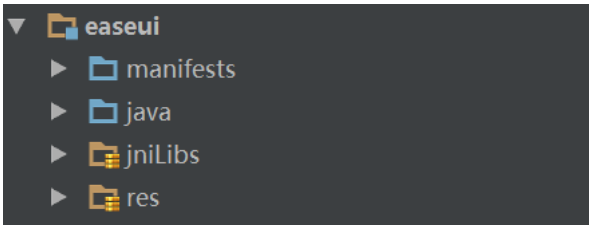
数据库选用的是 no-SQL 数据库 Mongodb。Mongodb 相比传统的 MySQL 有更灵活的数据存取方式和更好的性能。mongoDB 数据库的选择是因为我们这一产品的各个 model 之间的对应关系不是很复杂，而 mongoDB 灵活的特点，非常适合我们这个项目。

6.4 原创性说明

这整个项目是 2016 年 9 月份进行开发的，开发这个项目是为了参与当时的“花旗杯”金融应用创新大赛，当时参与比赛的虽然不止我们这个小组四个人，但是代码部分确实是由我们四个人完成的。

项目中的所有文档都是我们小组的人原创。代码部分只有客户端中为了使用“环信移动客服 SDK”引入了三个模块，这三个部分非原创，引用自环信 SDK 官方的 Demo。





7 附录

7.1 附加文档

（1）**项目开发计划：**描述项目开发的背景及目标、起讫时间、开发人员、开发过程规范和计划进度。

（2）**可行性研究报告：**分析并报告项目经济、技术、法律的可行性及开发方案的选择。

（3）**需求分析说明书：**软件系统在功能、行为、性能、设计约束等方面的信息，作为以后软件交付验收的依据。

（4）**数据要求说明书：**描述输入输出数据的，对数据的要求、范围、预处理等的采集。

（5）**数据库设计说明书：**详细规定了数据库的设计，包括数据表的概念结构设计、逻辑结构设计和物理结构设计。

（6）**概要设计说明书：**说明该软件的大概的设计的文档。

（7）**详细设计说明书：**将软件和数据的描述进一步精化成软件的算法表示和数据结构，规定各系统层次中每一个模块的设计细节。

（8）**测试计划：**制定为测试程序每一个模块及整个系统而设计的数据与测试工具，并提供测试的期望结果和准测尺度。

（9）**测试分析报告：**记录测试的结果及发现，分析系统功能的实现、缺陷与限制，并提出修改建议和评价。

（10）**项目开发总结报告：**综合描述项目产出的系统的主要功能和性能、程序的运行基本流程，总述开发的进度及费用，提出项目开发各方面的评价及总结经验教训。

（11）**软件问题报告：**指出软件问题的登记情况，如日期、发现人、状态、问题所属模块等，为软件修改提供准备文档。

（12）**源程序：**软件开发过程中的全部代码以及注释。

7.2 参考资料

1.“赢行家”个性化社交理财平台软件需求说明 来源：中山大学 Crazyfish 团队

2. 文档撰写的参考：计算机软件产品开发文件编制指南 （GB 8567-88）；

HB 6465-1990 软件文档编制规范

3.《软件工程－实践者的研究方法》（第六版）Roger S. Pressman 著，郑人杰、马素霞、白晓颖等译，机械工业出版社